HPI Hasso Plattner Institut
IT Systems Engineering
Universität Potsdam

Master Thesis

Language Identification using Deep Convolutional Recurrent Neural Networks

<Deutscher Titel bei englischer Arbeit>

Tom Herold

tom.herold@student.hpi.uni-potsdam.de

XX.0X.2017

Supverisors
Prof. Dr. Christoph Meinel
Dr. Haojin Yang

Internet-Technologies and Systems
Hasso Plattner Institute
University of Potsdam, Germany

Abstract

Zusammenfassung

Acknowledgments

I would like to express my gratitude to my supervisors Dr. Haojin Yang and Prof. Dr. Christoph Meinel. I want to thank them for giving me the opportunity to research this interesting topic, as well as for their guidance and advice. I especially want to thank Dr. Yang for his insights, support, and inspiration regarding the deep learning techniques used in my masters thesis. I would also like to thank my colleagues Georg Wiese, Christian Bartz, Tom Bocklich, Norman Rzepka, Christoph Sterz, and Johannes Jasper for the many fruitful discussions about language identification and machine learning in general. I owe them a great deal of gratitude for supporting and encouraging me while working on this thesis.

Thank you.

Contents

Abbreviations

ASR automatic speech recognition

CNN convolutional neural network

CRNN convolutional recurrent neural network

FC fully connected layer

GPU graphics processing unit

LID language identification

LSTM long short-term memory

NIST National Institute of Standards and Technology

RNN recurrent neural network

t-SNE t-distributed stochastic neighbor embedding

1. Introduction

TODO

1.1. Contributions

In this thesis we present an approach to language identification systems using deep learning techniques. We transfer the given audio classification problem into an image based task to apply image recognition algorithms. Our contributions can be summarized as follows:

- We investigate the suitability of convolutional neural networks for the task of language identification. We propose a hybrid network, combining the descriptive powers of convolutional neural networks with the ability of recurrent neural networks to learn time series. This approach is called convolutional recurrent neural network (CRNN).

- We implemented a CNN and a CRNN system in Python using the deep learning framework Keras and TensorFlow. We show that the CRNN approach outperforms all other methods in every single evaluation with respect to accuracy and F1 score.

- To train our system we gathered our own large scale dataset of audio recordings. We explain how we obtained and processed more than a thousand hours of human speech data suitable for our task.

- We assessed several machine learning metrics on our system with respect to our test data. Furthermore, we investigated the influence of noisy environments on our system. We discuss the system's ability to differentiate between several languages and how such a system can be extended to more languages.

- To showcase our system we developed a web service demo application using our best performing model. Further, we published said model for use by others.

1.2. Outline of the Thesis

This thesis is structured as follows: In chapter 2 we introduce the language identification problem and state our research hypotheses. Chapter 3 explains the theoretical background of the deep learning techniques and algorithms used in this thesis. Chapter 4 introduces related work and alternative approaches to the language identification (LID) task. In

chapter 5 we describe the audio datasets we collected for training and evaluation of our system. Implementation details are outlined in chapter 6. Further, we describe the network architectures of our models. Evaluation results are reported and discussed in chapter 7 and followed up by various experiments for assessing the music and noise robustness of our system. In chapter 8 we propose a web service to showcase a potential use case for language identification. Finally, we close this thesis by summarizing all our observations in chapter 9 and outline future work.

2. The Language Identification Problem

2.1. Language Identification as the key to speech tasks

Automatic language identification (LID) is the process of determining the language spoken in an audio recording. The language identification systems proposed in this thesis consumes audio recordings and uses deep learning techniques to do an automated classification of the recording's source language. Language identification is often the first step in a natural language processing pipelines. Automatic speech recognition (ASR) is the task of transcribing spoken language into readable text, sometimes also known as speech-to-text system. In contrast to LID systems, ASR uses a complex ensemble of multiple machine learning models, one of which is usually a language model. Selecting the correct language model is crucial to combine the transcription of single letters into meaningful words and correct grammar. Automatic language identification systems can serve a wide variety of applications from industry and research to entertainment.

Many call center operators benefit for LID systems by automatically routing telephone calls to connect a caller with a native speaker. This approach can be used both for customer care hotlines connecting callers with help desk agents as well governments agencies such as emergency telephone services. Muthusamy et al[1] report that manual matching emergency calls to US law enforcement agencies with native speaking agents can involve a significant delay of up to three minutes. Automatic language identification systems could speed up this process and efficiently support human agents.

Similarly to language identification some research is focused on dialect detection. To foster research in this field DARPA established the TIMIT[2] dataset for language identification of North American speakers. The dataset features either major North American dialects and has long been the default corpus for comparative research on lid systems. Recently, Germany's Federal Office for Migration and Refugees (BAMF) announced the plans for using dialect identification systems[1] as an additional resource in identifying a person's origin.

Language identification is also the first step to translation tasks. At the time of this writing we discovered that not even the Google Translate mobile app has automated language detection to determine the input language for a speech sample. Automated input language detection is available for written texts but Google Translate's voice input feature is only available for use after manually selecting an input language. We believe that having an automated language detection system could be extremely helpful to users.

An ever increasing number of people connect to the internet or communicate with each

[1] `http://www.theverge.com/2017/3/17/14956532/germany-refugee-voice-analysis-dialect-speech-software, accessed13.04.2017`

other using their smartphone. While touchscreen input can sometime be complicated the demand for voice interactions grow. Already many tasks can be solved through voice input. This hands free, voice enabled interaction is trend that we see in other industries such as cars as well.

Automatic LID systems and machine intelligence are also helpful for some entertainment companies. When we first started working on this thesis we were briefly inspired by the challenges of Berlin based startup Dubsmash[2]. Their app lets user's create a mashup of a large variety of existing songs, movie quotes or other voice snippets with a ten second video recording of the user. The resulting video clip can be shared with friends and usually features a funny and personal reinterpretation of the audio source's original context. The success of the app is directly proportional to user's interest in the offered sound clips. In some cases, however, users were offered clips in foreign languages which were often perceived as not funny or inappropriate. A LID system can be used to classify the language of the millions of audio snippets available in their library. Consequently, only sounds in the user's native language can be recommended to the user.

2.2. Task Specification in This Thesis

In this thesis we propose a language identification (LID) system for classifying the languages of a given audio recording. We make use of human voices recordings to train the system. We evaluate the suitability of convolutional neural networks for a LID system. We extend this approach with a recurrent neural network to form a hybrid network known as convolutional recurrent neural network (CRNN). The system 's performance is evaluated on a set of news broadcasts and speeches made by members of the European Parliament. We further assess the system's robustness to noisy environments and background music. We state the following hypotheses:

1. Convolutional neural networks can be successfully used for language identification tasks with high accuracy.

2. Spectrogram images are a suitable input representation for learning audio features.

3. Convolutional recurrent neural networks improve the classification accuracy for our LID task compared to a CNN based approach.

[2] `https://www.dubsmash.com/`, accessed 13.04.2017

3. Theoretical Background

3.1. Machine Learning

3.1.1. Types of Machine Learning

3.1.2. Classification

3.2. Building Blocks of Neural Networks

3.2.1. Fully Connected Layer

3.2.2. Convolutional Layers

3.2.3. Pooling Layers

3.2.4. Batch Normalization Layers?

3.2.5. Softmax Loss Function

3.3. Recurrent Neural Networks

3.3.1. Long Short Term Memory Networks

3.4. Hybrid Networks

- Convolutional Recurrent Neural Networks
- What is their purpose? Averaging over predictions / majority voting

3.5. Audio Representations

- MFCC
- Spectrogram (harmonics, formants)

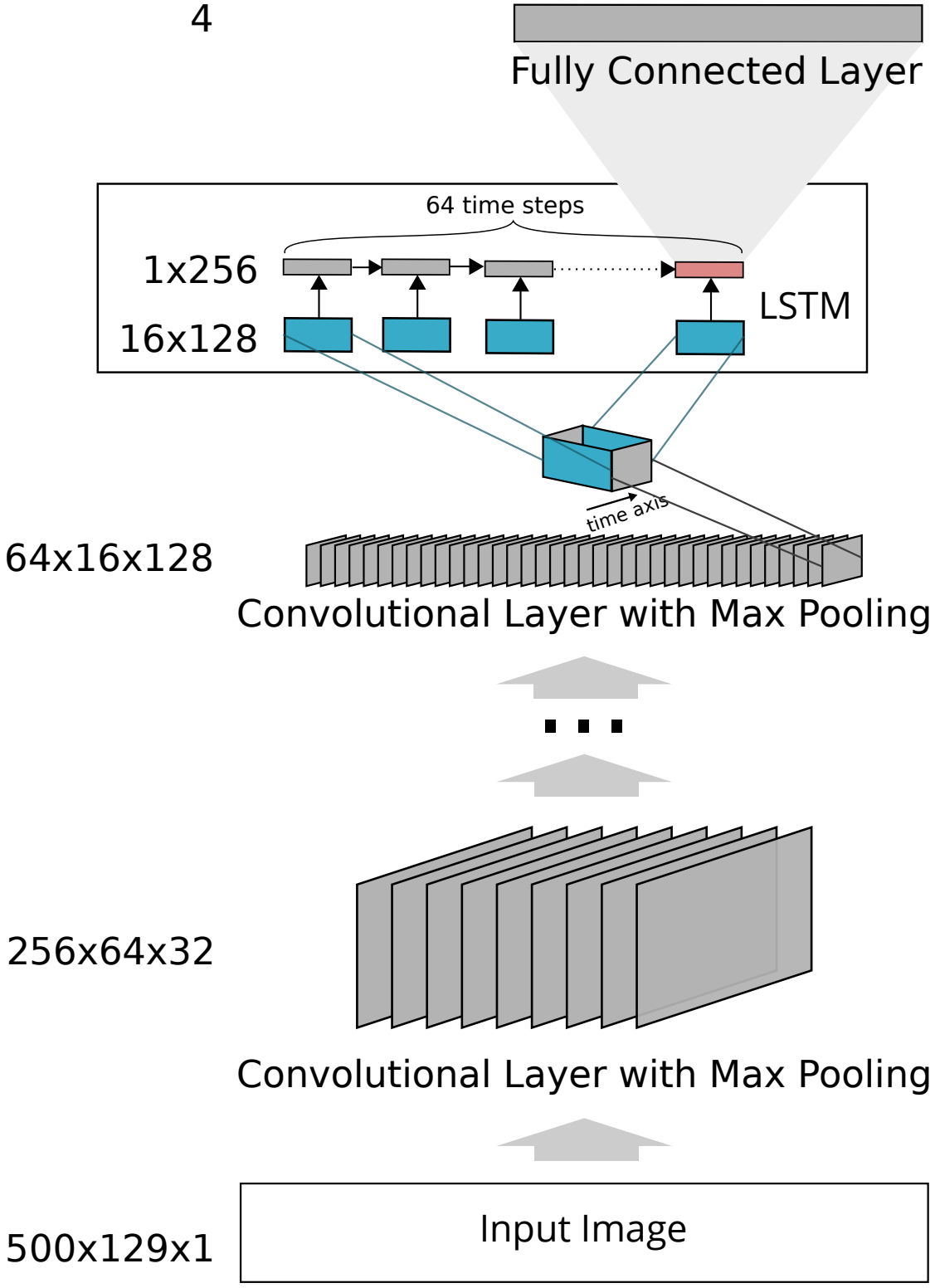

Figure 1: Our proposed CRNN hybrid network architecture consists of two networks. A CNN transforms our input images into an intermediary representation of our audio frequencies. The 3D output of final convolutional layer of the CNN is sliced along the x-axis (time axis) into 2D time steps still containing all feature map information. The output of the final LSTM time step is fed into a fully connected layer for classification.

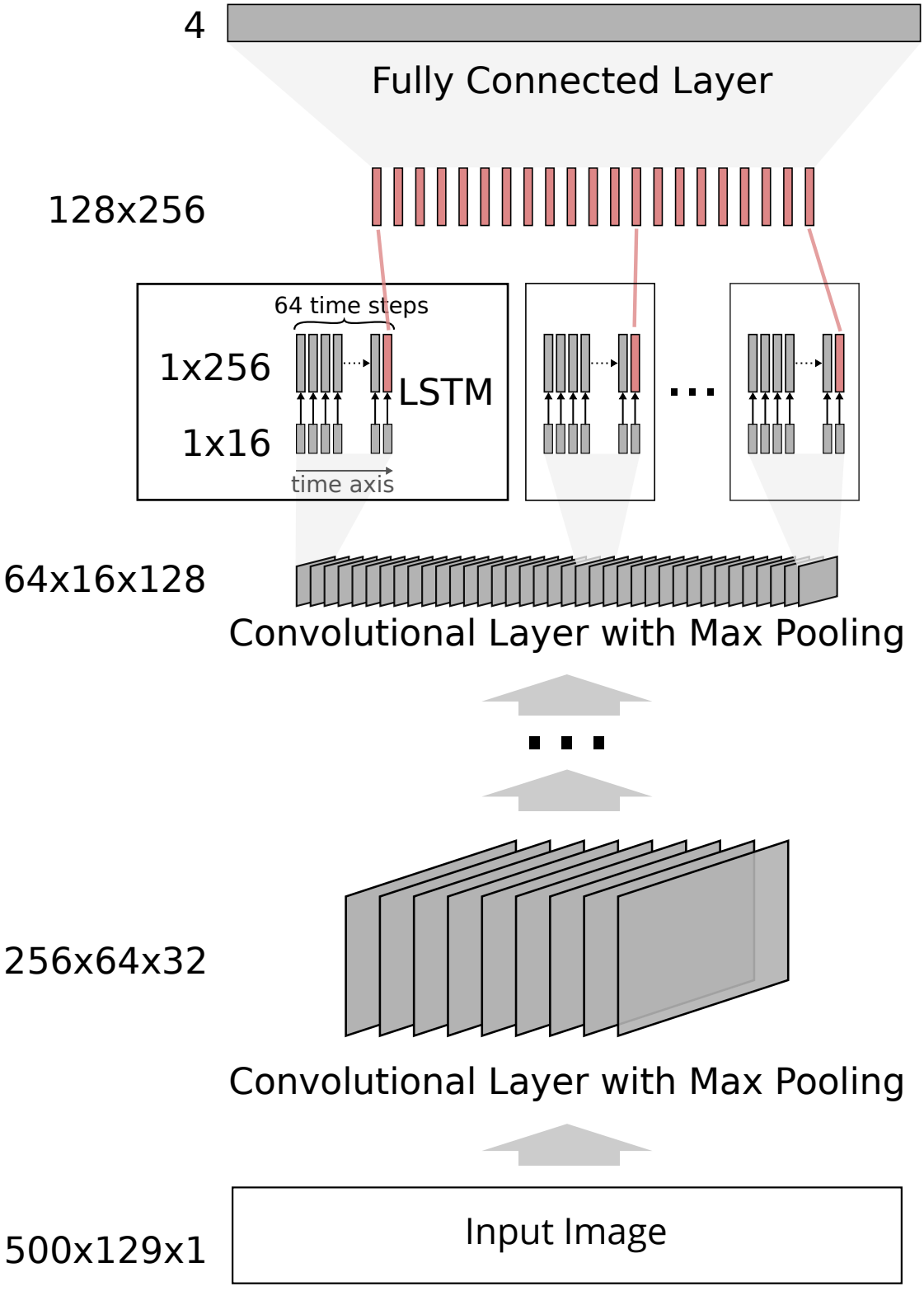

Figure 2: An alternative approach for a hybrid CRNN network. Each single feature map of the final convolutional layer is fed to separate LSTM networks as a 2D input. Each LSTM interprets the vector entries along the x-axis as time steps and operates on thin slice of the data. The output of the final time step of each LSTM is concatenated into a single vector serving as the input to a fully connected classification layer.

- https://home.cc.umanitoba.ca/ robh/howto.html
- Waveform
- Mel-scale
- frequency –> phoneme –> word –> sentence –> language

4. Related Work

4.0.1. A UNIFIED DEEP NEURAL NETWORK FOR SPEAKER AND LANGUAGE RECOGNITION

- [3]
- Almost identical: Deep Neural Network Approaches to Speaker and Language Recognition [4]
- high level overview of i-vector system
- Task: Language Recogniton + Speaker Recognition
- use bottleneck feature in the second to last layer
- input 7 static cepstra appended with 49 SDC
- DNN has 7 hidden layers of 1024 nodes each with the exception of the 6th bottleneck layer which has 64 nodes
- LRE11

4.0.2. EXTRACTING DEEP NEURAL NETWORK BOTTLENECK FEATURES USING LOW-RANK MATRIX FACTORIZATION

- [5]
- bottle neck feature improve classification results
- Task: Automatc Speech Recogniton (ASR)
- get bottleneck feature by low rank matrix factorization
- this is done by replacing the usual softmax layer weights by a linear layer with a small number of hidden units followed by a softmax layer
- BN laier is always last layer
- linear layer = FC without activation func

- uses DNN with 5 FCs with 1024 hidden units each + sigmoid activations + 1 BN layer
- softmax cross entropy loss
- 23 critical-band energies are obtained from a Mel filter-bank, with conversation-side-based mean subtraction = 150 dimensions
- further reduciton of output by PCA
- hybrid system of DNN + BN feeding into DNN + BN

4.1. LRE 2015

4.1.1. BAT System Description for NIST LRE 2015

- [6]
- participate in the "Fixed" and "Open" LRE Challenge
- segment data using automated Voice Activity Detection = previously trained NN
- 3042 segments (248 hours of speech) in train set and 42295 segments (146 hours of speech) in dev set.
- inputs 24 log Mel-scale filter bank outputs augmented with fundamental frequency features from 4 different f0 esti- mators
- used i-vector system

4.1.2. Discriminating Languages in a Probabilistic Latent Subspace

- [7]
- Probabilistic Linear Discriminant Analysis (PLDA) model
- In this paper, we review state-of-the-art generative methods, based on the Total Variability (TV) model [10], with the aim to improve their performance with discriminative fine-tuning of each language cluster at a time.

- TV maps audio into single low-dimensional vector, i-vector, that contains speaker, channel, and phonetic variability

4.1.3. Evaluation of an LSTM-RNN System in Different NIST Language Recognition Frameworks

- [8]
- used a one directional LSTM
- perform significantly better than i-vectors systems in LRE
- nice high level description of how LSTMs work
- inputs: random chunks of 2 seconds from which MFCC-SDC (Shifted Delta Coefficients)
- softmax cross entropy loss
- use last frame for scoring
- comparison if i-vector baseline to LSTM
- only used training data for 8 languages with more than 200hours of data
- US English (eng), Span- ish (spa), Dari (dar), French (fre), Pashto (pas), Russian (rus), Urdu (urd), Chinese Mandarin (chi)
- data split into 3, 10 and 30 seconds
- model: two hidden layers of 512 units followed by an output layer. The hidden layers are uni-directional LSTM layers while the output layer is a softmax with as many units as languages in the cluster
- LSTM os only better for short utterance (<= 10s)
- LSTM uses less parameters than i-vector

4.1.4. Frame-by-frame language identification in short utterances using deep neural networks

- [9]
- highlights the downsides/disadvantages of i-vector systems

5. Dataset Compilation

In this chapter we explain the structure of our datasets and how we obtained them.

Recent advances in deep learning were fueled by the availability of high performance hardware, especially massively parallel computing graphic processors units (GPU), and of large-scale, well-annotated, public datasets, for example ImageNet [10] for the computer vision domain. Historically, within the language identification community the TIMIT corpus of read speech [2] has long been the default test set. TIMIT contains a total of 5.4 hours, consisting of 10 sentences spoken by each of 630 speakers from 8 major dialect regions of the United States. All samples are recorded at 16kHz, which is a significantly lower quality than the 48kHz that are standard today . Given the short span of each individual sound clip, the overall corpus duration and restriction to only one language TIMIT was unsuited for this thesis. Therefore, it was necessary to obtain our data elsewhere.

This thesis uses two primary datasets collected and processed by us. We processed speeches, press conferences, and statements from the European Parliament and collected news broadcasts sourced from YouTube.

5.1. Language Selection

For the scope of this thesis we decided to limit ourselves to a number of languages spoken by many millions around the world. We focused our efforts on languages with a high availability of public speech content present in the various data sources explained below, namely the EU Speech Repository and YouTube. From a linguistic standpoint we also made sure to include language from within the same language family with similar phonetics for comparison reason. More on the similarities of related languages in section 7.4.5. Following these guidelines we decided on two Germanic languages, English and German, and two Romance languages, French and Spanish. We later extended our selection to Russian and Mandarin Chinese.

5.2. EU Speech Repository

The EU Speech Repository[3] is a collection of video resources for interpretation students provided for free by the European Commission. The dataset consists of debates of the the European Parliament as well as committee press conferences, interviews and tailor-made training material from EU interpreters. Each single audio clip is recorded in the speaker's

[3] `https://webgate.ec.europa.eu/sr/`, accessed 10.03.2017

native language and features exactly one speaker. Overall, however, the dataset consists of many different male and female speakers adding a nice variety to the data.

With 131 hours of speech data it is significantly smaller then the YouTube dataset. We obtained material in four languages: English, German, French, and Spanish. Prior to downloading the data, we gathered and processed every single webpage containing a single video in our target language by using the Selenium website end-to-end testing framework. We downloaded and extracted the audio channel of the source videos using the command line tool youtube-dl[4].

5.3. YouTube News Collection

Following the approach of Montavon [11] who used podcasts and radio broadcasts as input data we looked for large, public sources of speech audio. Both podcasts and radio stations have disadvantages for this thesis' language identification task for several reasons: Podcasts are usually restricted to one single speaker and lack variety. Radio, on the other hand, contains a lot less speech content and consists mainly of music and songs. Consequently we decided on using news broadcasts which provide high quality male and female speech audio data suitable to our needs. To obtain a large variety of languages and gather enough hours of speech audio we sourced the majority of our data from YouTube.

For each target language we manually selected one or more YouTube channels of respected news outlets, e.g. BBC and CNN for English. Using more than one channel for each language has the benefit of collection a wide variety of accents, speech patterns and intonations. For a full list of channels refer to table 1. All channels were chosen regardless of their content, their political views or journalistic agenda. Again, we downloaded the data using the command line tool youtube-dl and saved only the audio channel.

Audio obtained from news coverage has many desired properties. The data is of high recording quality and hundreds of hours recording is available online. News anchors are trained to speak natural and conversational, while maintaining at steady speed of about 150 words per minute[12]. News programs often feature guests or remote correspondents resulting in a good mix of different speakers. Unlike audio book recordings with texts being read aloud, news anchors converse in regular, human, conversational tone with each other. Lastly, news programs feature all the noise one would expect from a real world situation: music jingles, non-speech audio from video clips and transitions between reports. On the one hand this might improve the models's noise robustness. On the other hand this may interfere with the training, e.g. by having audio segments containing less than average length speech fractions. Although some broadcasts feature mixed language parts, e.g. foreign city, company, and personal names, we believe this is not a big problem.

[4] `https://github.com/rg3/youtube-dl`, accessed 23.03.2017

YouTube Channel Name	Language
CNN	English
BBCNews	English
VOAvideo	English
DeutscheWelle	German
Euronewsde	German
N24de	German
France24	French
Antena3noticias	Spanish
RTVE	Spanish
VOAChina	Mandarin Chinese
Russia24TV	Russian
RTrussian	Russian

Table 1: YouTube channel names used for obtaining the speech data and their corresponding language.

	EU Speech Repository	YouTube News	YouTube News Extended
Languages	English, German, French, Spanish	English, German, French, Spanish	English, German, French, Spanish, Russian, Chinese
Total audio duration	131h	942h	1508h
Average clip duration	7m 54s	3m 18s	4m 22s
Audio Sampling Rate	48kHz	48kHz	48kHz

Table 2: Comparison of our collected EU Speech Repository and YouTube News dataset. With about 1000 hours of audio recordings the acquired YouTube dataset is ten times larger than the EU Speech Repository.

The pronunciation and intonation of these words and phrases still follow the host's native language.[5] In essence we believe that speech data sourced from news broadcast represent an accurate, real-world sample for speech audio.

In contrast to our EU Speech Repository this dataset consists of ca. 1000 hours of audio for the same four languages: English, German, French and Spanish. Additionally, we also gathered an extended language set adding Mandarin Chinese and Russian. The extended set is only used for the evaluation of the model extensibility as outlined later. Table 2 provides a comparison between the two datasets.

5.4. Other Datasets

- TIMIT
- NIST LRE

Compare to related work datasets

[5] E.g. According to the international phonetic alphabet (IPA) the city of Berlin has many different pronunciations in different languages: [bɛʁˈliːn] (German), /bəˈlɪn/ (British English), and [bɚˈlɪn] (American English).

6. Implementation

This section outlines the software resources of our language identification system, explains the necessary data preprocessing, and describes the model architecture of our neural networks in more detail.

6.1. Software

Our language identification system is implemented in Python 3 and uses the open source deep learning framework Keras[13] with the TensorFlow[14] backend for training our neural networks. Keras provides us with a set of higher level machine learning primitives such as convolutional layers and optimization algorithms such as stochastic gradient descent without sacrificing any fine grained control over the training parameters. Internally it builds on Google's open source numerical operation library TensorFlow which is optimized to quickly compute multidimensional data on GPUs. We make heavy use Keras' aforementioned primitives, e.g. convolutional layers and the efficient LSTM implementation. The recently announced[6] version 1.0 of TensorFlow even advertises the ability to generate a small and efficient binary version of our models ready to be deployed on mobile phones.

All models are persisted to disk during training, including a summary of the layer architecture as well as their weights. This makes it easy to load, evaluate and deploy all models later. During the evaluation phase all performance metrics (accuracy, precision, recall, F1 score) are calculated using the Scikit Learn[15] framework. All measurements are logged and visualized using TensorBoard[7], both for the training and validation set. Having the ability to easily study and compare different metrics such as accuracy and loss across several training runs made it very comfortable to continuously monitor the progress of our research. Figure 3 shows a TensorBoard instance with metric plots for several models. We regularly compared different approaches and models against each other and used these plots to select the best performing systems for further experiments.

6.2. Data Preprocessing

All audio files undergo preprocessing before being fed to the neural network. As a first step all files are encoded as uncompressed, lossless Waveform Audio File Format[8], WAVE,

[6] `https://research.googleblog.com/2017/02/announcing-tensorflow-10.html?m=1`, accessed 03.03.2017

[7] `https://www.tensorflow.org/how_tos/summaries_and_tensorboard/`, accessed 30.01.2017

[8] `http://www.microsoft.com/whdc/device/audio/multichaud.mspx`, accessed 23.02.2017

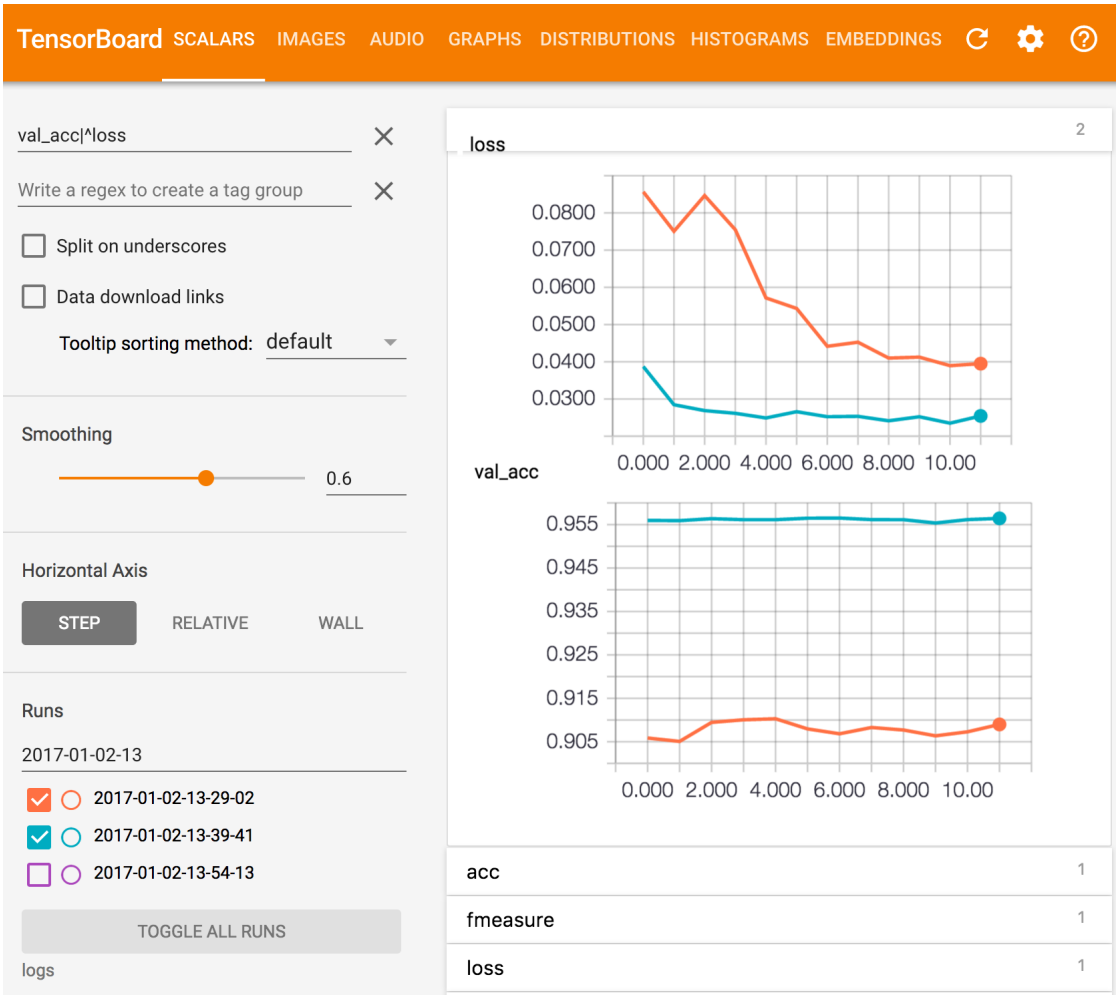

Figure 3: A TensorBoard instance showing training loss and validation accuracy of two different models. We regularly tried different approaches and model architectures and plotted multiple evaluation measures for comparison. This workflow enabled us to judge the impact of each experiment and measure the overall performance progress of the system.

commonly know by its file extension *.wav. This conversion has two advantages: A lossless data codec allows for future audio manipulations without any deterioration in signal quality and makes the data easily interchangeable for third party programs and libraries such as SciPy[16].

Since all our neural networks do not operate on raw waveform audio signals directly we transfer our features into the image domain. As introduced in section 3.5 we used a spectrogram representation of the audio file for training our models. The spectrograms were generated using the open source command line tool SoX[9]. The spectrograms are discretized using a Hann window[17] and 129 frequency bins along the frequency axis (y-axis) as instructed by SoX manual[10]. Human voice frequencies for male voices begin between 150Hz and 255Hz[18]. Female voice frequencies are usually shift by one octave and begin at 210Hz. Typically voice frequencies range up to 3.4kHz. For reference, the human ear is capable of recognizing frequencies from 20Hz to 20kHz with most sensitivity in the region of between 300Hz and 10kHz. Single sounds, however, exceed these limit significantly. Generally, voice frequencies are influenced by gender, age as well as various other factors such as the language, the type of discourse, and the emotional state of the speaker. Figure 4 shows the frequency areas with high energy in response to the tone of different vowels of human speech for the English language. Consequently, we instructed SoX to only include frequencies up to 5kHz into the spectrograms. The time axis (x-axis) is rendered at 25 pixel per second. Each audio sequence is clipped into non-overlapping ten second segments. The final segment is discarded to avoid segments shorter than the required ten seconds to avoid padding. Since we gathered enough training data we decided against padding with black pixels, which could be interpreted as silence and add unnaturally long speech pauses. We also decided against filtering silent sections within the 10 second audio segments to keep the natural pauses between words and not disturb the regular speech rhythm. Frequency intensities are mapped to an eight bit grayscale range. We combined all audio channels into a single mono channel in order to generate only a single spectrogram image. The resulting grayscale images are saved as lossless PNG files with 500 pixel in width and 129 pixel in height. Listing 1 shows the SoX command for generating a spectrogram image from an input audio file.

[9] `http://sox.sourceforge.net/`, accessed 23.02.2017
[10] `http://sox.sourceforge.net/sox.pdf`, page 32, accessed 26.03.2017

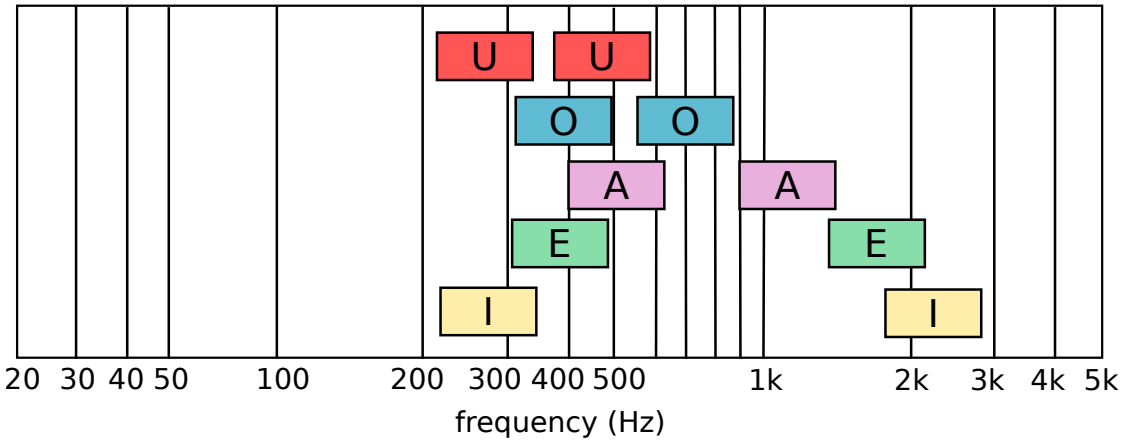

Figure 4: The upper and lower frequency areas (formant) for different vowels. These frequency ranges are typical for human speech. The lower speech formant f1 has a total range of about 300Hz to 750Hz and the higher speech formant f2 has a total range of about 900Hz up to over 3000Hz. But each single spoken tone has a much narrower range for both formants.

```
1
2     sox -V0 input.wav -n remix 1 rate 10k spectrogram -y 129 -X 50 -m -r -o
          spectrogram.png
3
4
5     V0 - verbosity level
6     n - apply filter/effect
7     remix - select audio channels
8     rate - limit sampling rate to 10k; caps max frequency at 5kHz according
          to Nyquist-Shannon sampling theorem
9     y - spectogram height
10    X - pixels per second for width
11    m - monochrome output
12    r - disable legend
13    o - output file
```

Listing 1: SoX command and options used for generating monochrome spectrograms. All audio files were discretized into 129 frequency buckets using a constant pixel width per time step resulting into spectrogram images of 500x129 pixels.

As seen in figure 5 the spectrograms feature very apparent bright ripple-like pattern. Each of these represents a strong activation of a certain frequency at a point in time. Several frequency activations can be active simultaneously constituting a particular phoneme or sound. A sequence of these phonemes forms words and is only interrupted by short speech pauses. We hypothesize that our LID system will learn the characteristical and unique composition of these frequency activation for every language in our classifier.

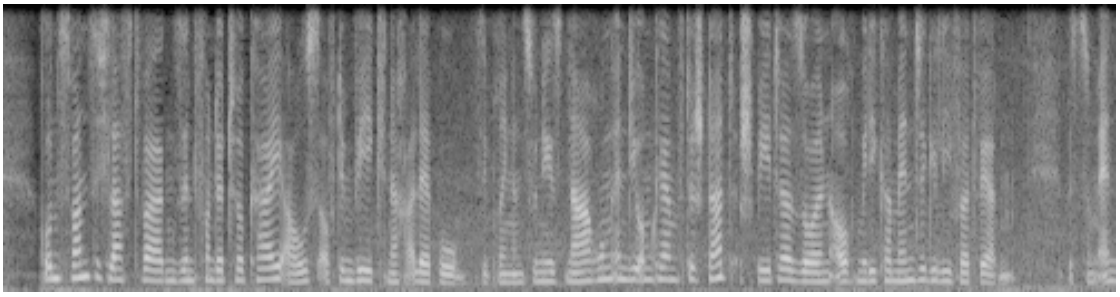

Figure 5: A spectrogram generated from a ten second German audio clip using SoX. Notice the bright ripple-like patterns representing high frequency activations. We hypothesize that these frequency activations will serve as the main features for the classifier.

6.3. Neural Network Architectures

To successfully solve our research task with deep neural networks we had to design a fitting model architecture. Combing the right layers and adjusting the correct parameters is not a trivial task. Critics argue that deep learning systems are a black box and that the impact of the individual layout of the network layers is hard to understand. Therefore, we based our model architectures on proven existing designs from related work and adapted them to our needs, while heeding as best practices[19, 20].

The architecture and the number of parameters of a deep neural network suited for a given task is determined by the bias-variance tradeoff[21]. It is the goal of a network's author to avoid underfitting or overfitting on the resulting model. The bias characterizes the prediction error that results from the model's limited ability to learn relevant relations of features in the training data. Underfitting can be caused by designing too small or too few network layers causing a high bias. Variance in contrast refers to the error that results from the model responding overly sensitive to variation in the training data. By designing a too large network and Introducing too many parameters or adding too many layers results in a model with low variance. Hence, it learns an exact representation of the training data without abstracting for more general applications. This is known as overfitting. Designing and adjusting the network layout is an iterative process. We tried many variations and parameters of our proposed model layouts to find the most suitable design for our LID system. The layout of the network interdepends and interacts with other design decisions of the system, especially the loss calculation. Each change in parameters involves a complete new training run to judge the effect of the change. Hence, the architecture always needs to be tested in its entirety.

For this thesis we tried three different design of convolutional neural networks. The first one and second one are based on the early VGG-style CNN-M model architectures of Simonyan et al. [22]. The CNN features five convolutional layers and modest feature map

output sizes. Note that the network's first two convolutional layers have comparatively large kernel size of 7×7 pixels and 5×5 pixels, respectively, yielding a large receptive field. Each convolutional layer is followed by batch normalization[23], a technique that helps in increasing training speed and achieving more model regularization. Following these we added a 2×2 pooling with a stride of two. After the five convolutional layers we added regularization through a fifty percent dropout before flattening all parameters to a fully connected layer with 1024 outputs. The final fully connected layer serves as a classifier outputting the prediction for our language identification. Henceforth, we will refer to this model as CNN_A. The full network layout can be seen in table 3.

A slightly adapted version of CNN_A has the same amount of convolutional layers but features a reduced number of feature maps. Instead of doubling the initial value of sixteen feature maps for every convolutional layer we stick to a schedule of 16 - 32 - 32 - 64 - 64 feature maps, respectively. The fully connected layer has also been reduceed to only 256 output units. Overall this model has significantly less parameters than the CNN_A. The purpose of this variation is to ensure that the proposed architecture for CNN_A is not unnecessarily complex. We called this variation CNN_B. The full network layout can be seen in table 3.

Lastly, we evaluated architecture CNN_C which is based on the VGG-16 network[24] and uses constant kernel size of 3×3 for all convolutional layers. At the same time we increased the number of convolutional layers to seven and extended the number of feature map outputs for each layer: 64 - 128 - 256 - 256 - 512 - 512 - 515. All fully connected layers are identical to the CNN_A. The main difference of this network design is the smaller kernel sizes and hence a smaller receptive field of the convolutional layers. To compensate for this we increased the number of layers. The complete CNN_C architecture is laid out in table 4.

For our CRNN hybrid network we constructed a convolutional neural network followed by a recurrent neural network. Specifically, we used a bidirectional long-short term memory network for the RNN part. We decided on using a bidirectional model of two LSTMs instead of a single LSTM based on the successful results of Shi et al[25]. The CNN part of the network is tasked with extracting features and providing a high dimensional intermediate frequency representation. The RNN part interprets every vector entry along the time axis of this intermediate interpretation as a single time step. For the CNN part we repurposed the CNN_A network architecture of five convolutional layers with larger kernels for the first two layers since we found this CNN layout to work best for our LID task. (Details are provided later in section 7.4.4.) We also kept the batch normalization and max pooling layers. The bidirectional LSTM layer consist of two single LSTMs with 256 output units each. We concatenated both outputs to a vector of 512 dimensions and fed this into fully connected layer with 1024 output units serving as the classifier. The complete model architecture can be seen in table 5.

We decided on training both parts of the hybrid network separately. First, we trained the

Layer Type	Output Size		Kernel	Stride
	CNN_A	CNN_B		
Convolution	16	16	7×7	1
Max Pooling	16	16	2×2	2
Convolution	32	32	5×5	1
Max Pooling	32	32	2×2	2
Convolution	64	32	3×3	1
Max Pooling	64	32	2×2	2
Convolution	128	64	3×3	1
Max Pooling	128	64	2×2	2
Convolution	256	64	3×3	1
Max Pooling	256	64	2×2	2
Dropout & Flatten	3328	832		
Fully Connected	1024	256		
Fully Connected	4	4		

Table 3: The layerwise architecture for the convolutional neural network CNN_A and CNN_B. This design is based on the early VGG style networks and features large kernel size for the first two convolutional layers in an effort to capture a large receptive field of features.

CNN part by itself just like in the previous section. This simplified the task of learning frequency features for the network and saved us some time. In practical terms this meant we could reuse the model weights of the previously trained CNN_A model. For the RNN part, we used a finetuning approach. This means we froze all convolutional layers weights by disallowing any further weight updates during the tracing phase. The bidirectional LSTM and FC layers, however, remained unfrozen and continued to be subject to weight updates and hence was trained regularly. Later, we also experimented with unfrozen RNN weights meaning we trained and updated both the CNN and LSTM layers at the same time. This approach, however, yielded worse results and we did not pursue it further.

why not automatic design?

why not rectengular kernels?

Layer Type	Output Size	Kernel	Stride
Convolution	64	3×3	1×1
Max Pooling	64	2×2	2×2
Convolution	128	3×3	1×1
Max Pooling	128	2×2	2×2
Convolution	256	3×3	1×1
Convolution	256	3×3	1×1
Max Pooling	256	2×2	2×2
Convolution	512	3×3	1×1
Convolution	512	3×3	1×1
Max Pooling	512	2×2	2×2
Convolution	512	3×3	1×1
Max Pooling	512	2×2	2×2
Flatten	6144		
Fully Connected	1024		
Fully Connected	4		

Table 4: The layerwise architecture for the convolutional neural network CNN_C. With seven convolutional layers this network design is deeper than the other two proposed CNN architectures. Additionally, the number of feature maps were increased to 512 units compared to the 256 units of the CNN_A network. Overall this network is deeper and consists of a higher number of parameters than the other two designs.

Layer Type	Output Size	Kernel	Stride
Convolution	$123 \times 494 \times 16$	7×7	1×1
Max Pooling	$61 \times 247 \times 16$	2×2	2×2
Convolution	$57 \times 243 \times 32$	5×5	1×1
Max Pooling	$28 \times 121 \times 32$	2×2	2×2
Convolution	$26 \times 119 \times 64$	3×3	1×1
Max Pooling	$13 \times 59 \times 64$	2×2	2×2
Convolution	$11 \times 57 \times 128$	3×3	1×1
Max Pooling	$5 \times 56 \times 128$	2×2	2×1
Convolution	$3 \times 54 \times 256$	3×3	1×1
Max Pooling	$1 \times 53 \times 256$	2×2	2×1
Transpose	$53 \times 1 \times 256$		
Reshape	53×256		
Bidirectional LSTM	1024		
Fully Connected	4		

Table 5: The layerwise architecture for the convolutional recurrent neural network. The network consists of two parts, a CNN and a bidirectional LSTM. It shares the same CNN architecture as the previously introduced CNN_A. The final convolutional layer is sliced into time steps along the x-axis (time axis) and serves as input to LSTM.

7. Experiments and Evaluation

In this chapter we present and discuss the results of training the outlined neural network architecture for spoken language identification. We assess several performance metrics evaluated on our system. Further, we experiment with modified model architectures to maximize the accuracy, improve its noise robustness and study the effect of background music on the neural network. To assess the real world performance of the the LID system we augment our data to simulate various noisy environments. Lastly, we show the classification performance of our approach and discuss the system's inter language discrimination and extensibility to other languages.

7.1. Hardware Resources

In order to facilitate Keras' and TensorFlow's hardware-accelerated computation we executed all trainings on CUDA[11] compatible GPU machines at our disposal at the Internet Technologies and Systems chair. Details can be found in table 6.

	Machine A	Machine B
OS	Ubuntu Linux 14.04	Ubuntu Linux 16.04
CPU	Intel® Core™ i7-4790K @ 4GHz	AMD FX™-8370 @ 4GHz
RAM	16GB	32GB
GPU	Nvidia GeForce® GTX 980	Nvidia Titan X
VRAM	4GB	12GB
Images / sec		

Table 6: Hardware resources used in training the neural networks. We made heavy used of modern GPUs to benefit from quick hardware-accelerated numerical computations.

measure images / sec

[11] `https://developer.nvidia.com/cuda-zone`, accessed 30.01.2017

	European Speech Repository	YouTube News
Training Set	18.788	193.432
Validation Set	5.372	55.272
Test Set	2.684	27.632
Total	26.844	276.336

Table 7: The amount of samples for our training (70%), validation (20%) and testing (10%) set for the respective datasets.

7.2. Data

For our performance evaluation we used the European Speech Repository and YouTube News dataset as described in chapter 5. Both datasets were preprocessed and converted to spectrogram images as described in section 6.2. Each spectrogram image represents a non-overlapping ten second duration of source audio file. We chose this duration in reference to the NIST LRE2015 challenge[26]. We split both datasets into a training (70%), validation (20%) and testing set (10%) and all files were distributed equally between all four language classes. The number of samples per class was limited by the language with the least amount of files to ensure an equal class distribution. The European Speech repository yielded a total of ca. 19.000 training images which amounts to roughly 53 hours of speech audio. The YouTube News dataset is considerably larger and yields a total of about 194.000 training files, or 540 hours. Table 7 contains the detailed dataset splits.

Given the European Speech Repository's smaller size we only used it initially to confirm the validity of our approach. At the point at which we were satisfied with the results we did not include it in the extensive robustness tests that we used for the evaluation on the YouTube News dataset. Moving on to a bigger challenge, we augmented the original audio of the news dataset with three different background noises to evaluate how well our model would hold out in non ideal, real world situations outside of a news broadcasting studio. For the first experiment we added generic white noise to data. For the second experiment we added noise to simulate an old phone line or bad internet connection during voice chat. Lastly, we added background music to the data. All experiments are described in detail below.

7.3. Training of the Neural Network Model

Neural networks have a multitude of hyperparameters that influence the training results drastically. In this section we will briefly explain our choice of hyperparameters alongside other important training settings:

Optimizer We used Adaptive Moment Estimation (Adam)[27] as our optimization method to quickly and efficiently reach convergence of our model. Adam utilizes momentum during gradient updates to support a quicker convergence. We set the optimizer's parameters β_1 to 0.9, β_2 to 0.999, and ϵ to 1e-08. For the majority of trainings we used Adam and only resorted to SGD during finetuning when we needed more control over the learning rate schedule and wanted smaller weight updates.

Weights Initializer All layer weights are initialized within the range (0, 1) using a normalized Glorot uniform initializer[28], also know as Xavier initializer. This heuristic is designed to compromise between the goal of initializing all layers to have the same activation variance and the goal of initializing all layers to have the same gradient variance. This initialized the biases to be 0 and the weights W at each layer with the following heuristic[29, chapter 8.4, p. 303]: $W \sim U(-\frac{6}{\sqrt{n_j+n_{j+1}}}, \frac{6}{\sqrt{n_j+n_{j+1}}})$, where $U[-a, a]$ is the uniform distribution in the interval $(-a, a)$ and n is the size of the previous layer.

Data Normalization The grayscale images are loaded using SciPy and normalized to value in the range of [0, 1]. The shape for all inputs needs to be uniform across all samples and is set to [500, 129, 1], unless otherwise noted. Data loading is handled through Python generators[12] to keep the system's memory footprint low.

Learning Rate We set the initial learning rate to 0.001. Given the Adam optimizer's dynamic learning rate adaption we expect the learning rate to be automatically increased or decreased after every iteration. Therefore, we did not specify a manual learning rate decay schedule.

Batch Size We specified the batch size depending on the available VRAM of the training machine. We used a value of 64 images per batch for Machine A and 128 images per batch for Machine B. (See section 7.1 for the hardware specifications.)

Weight Regularization Weight Regularization comprises any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error. We employed the L^2 norm[29, chapter 7.1.1, p. 231], sometimes also known as weight decay, as a weight regularizer for all convolutional and fully connected layers to improve the models generality. This regularization strategy

[12] `https://docs.python.org/3/glossary.html#term-generator`, accessed 30.01.2017

drives the weights closer to the origin by adding a regularization term to our loss function. We penalize our loss with a weight decay value of 0.001. Additional regularization happens through the use of batch normalization layers.

Epochs We limited the model training to a maximum of 50 epochs when using the Adam solver. We usually reached convergence well below this threshold. For training sessions with SGD we increased this parameter considerably since the weight updates per epoch are smaller and more training iterations are needed to reach convergence. To speed up our workflow we employed an early stopping policy and stopped a training if the validation accuracy and loss did not increase/decrease within a ten epoch window.

Metrics We observed the loss, accuracy, recall, precision, f1 measure for both the training and validation set during model training. All values were saved to log files and visualized as graphs in TensorBoard. A complete summary of all assessed metric can be found in section 7.4.1.

Loss As is common for multivariate classification all models were trained with a softmax cross-entropy loss function.

explain more. in theoretical background. add ref to section

7.4. Evaluation

7.4.1. Evaluation Metrics

In this section we discuss the evaluation metrics used throughout our experiments. All metrics are generally only defined for binary classification scenarios. Given our multi-class classification problem we will report the average of the individual class performance measures in the following sections.

Accuracy is a common measure in machine learning and is defined as the ratio of correctly classified samples to all samples in the dataset. In the context of language identification this translates as:

$$\text{accuracy} = \frac{\left|\{\text{correctly identified language samples}\}\right|}{\left|\{\text{all language samples}\}\right|}$$

Precision and Recall Precision defines the ratio of retrieved language samples that are correctly identified as belonging to said language. Recall is the fraction of correctly identified language samples to all samples belonging to this language. Both measures are calculated using true and false positives as well as false negatives. These are

defined as follows. *True positives* are all samples belonging to one language which were correctly identified as belonging to said language. In contrast, *false positives* are samples belonging to a language which were identified as belonging to another language. Lastly, *false negatives* are the samples belonging to a language which were incorrectly identified as not belonging to said language.

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

The F1 Score is the scaled harmonic mean of precision and recall. It is used to a have combined judgement of recall and precision, since it is generally not interesting to assess one without the other.

$$\text{F1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

7.4.2. Results for EU Speech Repository Dataset

In order to verify our theoretic model of using convolutional neural networks for classifying audio data in the image domain we established a baseline with the smaller EU Speech Repository dataset. Previous work with CNNs showed that the careful arrangement of the neural network layers has a great effect on the final classification performance. If one does not use enough or sufficiently large layers the model is not able to properly distinguish between classes. Going too deep or using too large layer outputs increases the overall number of model parameters to a degree where both training time and classification performance suffers again. The goal of this experiment is to find favorable settings for the number of convolutional layers needed, the kernel size of the convolutions, the number of output maps of the convolutional layers and finally the number of features of the fully connected layer.

In section 6.3 we introduced three proposals for different CNN architectures. CNN_A features large kernel sizes for the first two layers and we expect it to capture more information up front. A slightly smaller version of this design with less overall parameters is called CNN_B. It served as an assertion to verify that CNN_A's number of output feature maps were properly sized and not too large. Lastly, CNN_C features equally sized kernels and boasts both more convolutional layers as well as an increased number of feature map outputs.

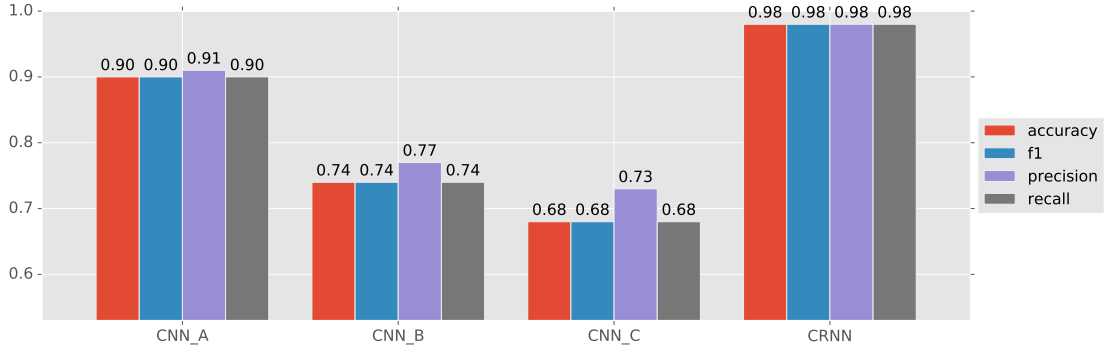

Figure 6: Performance measure comparison of three different CNN architectures evaluated on the EU Speech Repository dataset and our proposal of a CRNN model. CNN_A outperforms all other CNN approaches with a top accuracy of 0.90, but is bested by the CRNN's 0.98 accuracy, showing the potential of this thesis' approach.

CNN_A outperforms both of the other two network architectures with respect to all the evaluated performance measures, cf. figure 6. With a top-1 accuracy of 0.904 it trumps CNN_B and CNN_C with 0.743 and 0.68, respectively. Comparing the F1 score we get a similar result: 0.90 versus 0.74 and 0.68. This experiment confirmed a few of our initial hypotheses. Firstly, it proves that convolutional neural networks can be successfully used to classify audio data. Secondly, it demonstrates that spectrogram images are meaningful representations for audio that retain enough information for language identification. Thirdly, it shows that larger kernels for the initial convolutional layers are indeed favorable since the increased receptive field captures both the time and frequency domain better.

Based on these findings we did some further testing with CNN_A. Mishkin et al.[19] were able to gain a small improvements in accuracy by exchanging convolutional layer's activation function. Therefore, we replaced our Rectified Linear Unit (ReLU) activations to Exponential Linear Units[30] (ELU) but were unable to measure any improvement (0.85 accuracy). Baoguan et al. [25] proposed to use 1x2 rectangular pooling windows instead of the conventional square ones. This tweak yields feature maps with larger width, hence longer features along the time domain. In theory this should increase the CNN's sensitivity at capturing the occurrence of frequencies at certain time intervals. For this experiment we were unable to gain any improvement (0.81 accuracy), but we will come back to this technique for our CRNN approach later.

The task of this thesis was to evaluate the use of Deep Convolutional Recurrent Networks for language identification. Therefore we extended our previously best performing CNN_A with a bidirectional LSTM layer to capture time steps both going forward from the start and going backward from then end of an audio segment. We interpreted the CNN output

as intermediate high dimensional representation of the audio frequencies and used every vector entry along the x-axis as a single step / input for the LSTMs as explained in section 3.4. Confident in the features captured by our convolutional layers, we froze them during the CRNN training to disable any further weight updates for these layers. Instead, we focussed on only training the LSTMs weights and learn the frequency sequence of the audio sample. Our bidirectional LSTM layer trained two individual LSTMs with 512 outputs each, one training the input sequence from the start and one from the end. Both outputs were concatenated to form a single output vector with a dimension of 1024 which is followed by single fully connected layer for classification. Our CRNN architecture outperformed all CNN approaches significantly. With a top-1 accuracy of 0.98 and a F1 score of 0.98 it proves the viability of the CRNN approach and reaffirms the central hypothesis of this thesis.

7.4.3. Effect of Audio Duration

In all previous experiments we split the audio recordings into ten second segments, which translated to an image dimension of 500x129 pixels for the spectrogram. We decided on 10 second audio snippets to replicate on the setup of the NIST LRE 2015 [13] challenge. To study the effect of the audio duration on classification performance we set up two versions of the EU Speech Repository dataset with non-overlapping five and twenty second snippets but left the number of frequency bins unchanged. Hence, we changed the input dimensions to 250x129 pixels and 1000x129 pixels, respectively, and could not just use our previously trained models but had to retrain new models.

To set a baseline we used the same CNN_A architecture as explained in the previous section. When completely training the five second version from scratch we achieved an accuracy of 0.81 falling short of the results achieved with ten second snippets. Next, we applied some transfer learning and finetuned CNN_A on the new five second dataset. Since the convolutional layers are not bound to a specific input size and given that the frequency features did not change in dimension we were able to reuse the complete convolutional weights. For the finetuning we froze the convolutional layers, confident in their ability to detect frequency features, and only retrained the final two fully connected layers. With the bisection of the input data the amount of model parameters were greatly reduced, especially the fully connected layer weights. To account for this we finetuned one model with a fully connected layer of 512 outputs and a second one with the default 1024 outputs. Overall this yielded an accuracy of 0.88 and 0.89, respectively. We concluded that the effect of the smaller fully connected layer is only marginally better.

After establishing a solid CNN foundation we applied the same CRNN approach as previously highlighted. Due to the shorter audio duration the final pooling layer only

[13] `https://www.nist.gov/itl/iad/mig/2015-language-recognition-evaluation`, accessed 15.02.2017

Model Architecture	Accuracy	F1
CNN (5s) from scratch	0.81	0.81
CNN (5s) finetuned with 1024 Fully Connected Units	0.88	0.89
CNN (5s) finetuned with 512 Fully Connected Units	0.89	0.89
CNN (20s) from scratch	0.90	0.90
CRNN (5s) with 5 time steps	0.90	0.91
CRNN (5s) with 22 time steps	0.90	0.91
CRNN (10s) for reference	0.98	0.98

Table 8: Various CNN and CRNN model configurations trained on five second audio samples. The best performing five second CRNN still falls short of its ten second counterpart with an accuracy of 0.90 and 0.98, respectively.

features 5 output units along the x-axis compared to the 13 output units of the ten second CRNN. When interpreted as a sequence of time steps and fed into the bidirectional LSTM the accuracy improved only marginally to 0.90. We suspect that the number of time steps was too little to take full advantage of the recurrent network layer. In an effort to increase the number of output units of the final pooling layer and hence increase the sequence length we applied 1x2 rectangular pooling tweak again. We changed the final two pooling layers and increased the number of output units along the x-axis to 22. The y-axis remained unaffected. The resulting accuracy of 0.90 and the F1 score of 0.91 remained comparable to the previous model and did not bring the desired effect.

For the twenty second version we trained the network from scratch using the CNN_A architecture. This yielded an accuracy of 0.90, which is comparable to its ten second counterpart. This experiment uses double the amount of pixels along the x-axis as the ten second baseline. This expansion of data led to an increase in computation time. At the same time, we felt that our LID system is more flexible when requiring shorter input audio snippets rather then longer one. We concluded that these longer snippets did neither improve our accuracy nor did they make the LID system more attractive overall.

In summary we believe that both decreasing and increasing the duration of the audio snippets used for training has a negative effect on the classification performance. While it is possible to train and finetune CNNs that match their ten second counterparts we found that the CRNN approach does not boost the model effectiveness in a similar manner.

7.4.4. Results for YouTube News Dataset

Following the promising results from the experiments with the EU Speech Repository we switched to the ten times larger YouTube News dataset for further training and evaluation. For our first experiment we used the same CNN_A architecture as before but initialized the model weights with the weights of best performing model of the EU Speech Repository evaluation. Our reasoning here was to reuse the convolutional layers that were already trained to identify frequency ranges. However, with an accuracy of only 0.79 it did not perform as strongly as anticipated. One reason for this could be that the EU dataset is a lot smaller and does not feature as many diverse situations as present in news broadcasts. In the case of broadcasts all audio is recorded in a similar environment without much background noise and exhibits high signal quality.

Next, we trained the same CNN completely from scratch with randomly initialized weights. We had several setbacks when using an Adam optimizer and reverted to using standard SGD to keep our loss in check. Specifically, we encountered the exploding gradient problem[29, ch. 8.2.4, p. 288] and had to use gradient clipping to solve the issue. With this tweak we were able to get the model to converge and gained an accuracy of 0.9090 besting our previous attempt. Given the larger size of the new dataset we also tried to increase the number of parameters by doubling the feature maps of the convolutional layers. This, however, did not help. As a next step we applied the CRNN approach again. Based on this CNN we added our bidirectional LSTM layers in the same manner as for the previous CRNNs and were able to slightly improve our accuracy and F1 score to 0.9124, respectively.

To evaluate how our model architecture fares against established deep learning models we trained a model using Google's Inception-v3[20] layout. This network design features more layers and is considerable deeper than our proposed CRNN architecture and is the result of Google's latest research into neural networks for image recognition tasks. Instead of a monolithic network design of sequential layers it uses a combination of parallel inception units, a sequence of convolution layers as one building block. Evaluating the Inception-v3 model resulted in a top accuracy of 0.9488 improving our results significantly. Applying the established CRNN treatment to this model increased the performance by roughly 1% to an accuracy of 0.9579. In order to boost the performance even further we also tried a CRNN variation where both the convolutional layer weights and the LSTM weights were trained. In contrast to our initial CRNN approach this method also updates the existing convolutional filter. We found, however, that this variation did not improve performance. Figure 7 shows an overview of the performance metrics of the mentioned experiments. The increased performance, however, does not come without a cost. With a total of 3.153.924 parameters our CRNN uses roughly six times less parameters then the Inception-v3 CRNN with its 19 million. This increases training time, requires more training data and consumes more GPU memory. On disk the serialized model weights come in at 30MB versus 260MB, which could be a potential disadvantage for deployment

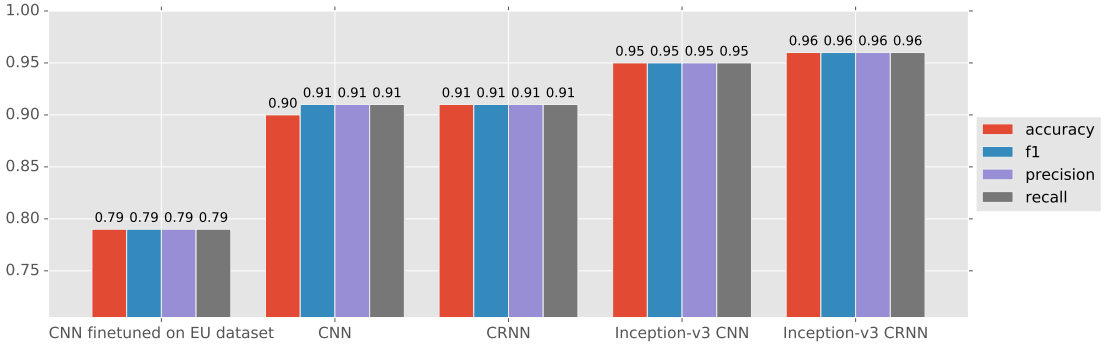

Figure 7: Performance measurement comparison between our CNN and CRNN models and Inception-v3 based models. With a top accuracy of 0.96 the Inception style CRNN performs best but needs more than five times the amount of parameters compared to our proposed model architecture.

on mobile phones.

7.4.5. Inter Language Discrimination

Previous work[11] raised concerns about the similarity of our four feature languages – English, German, French and Spanish – and the model's inability to discriminate between them properly. Both English and German belong to the West Germanic language family, while French and Spanish are part of the Romance languages. We hypothesized that our deep learning approach to language identification is able to differentiate between them reliably. Table 9 shows the confusion matrix when evaluating language family pairs on the best performing CRNN. Spanish and French audio files separate very well with hardly any wrong classifications. Both languages are more likely to be classified as German and English rather then as the respective other, demonstrating that their learned representations are quite distinctive within their language family. German and English language samples have a tendency to be misclassified as the respective other. Furthermore English also has a slight bias towards French, an observation in line with related work[31]. German, however, distributes its classification error evenly between French and Spanish. Overall, German samples are misclassified the most across all languages.

The confusion matrix for the Inception-v3 CRNN, table 10, match our observations for the other model. Again, German exhibits the most classification errors.

	EN	DE	FR	ES
EN	6153	339	181	225
DE	426	6128	173	162
FR	200	145	6447	107
ES	214	170	115	6399

Table 9: Confusion matrix of our best performing CRNN. Ground truth values are along the row-axis. Predicted values are along the column-axis. English audio files are likely to be misclassified as German and vice versa. Both languages belong the family of Germanic languages.

	EN	DE	FR	ES
EN	6648	140	45	61
DE	152	6639	62	44
FR	68	59	6742	27
ES	75	83	42	6697

Table 10: Confusion matrix of the Inception-v3 CRNN. Ground truth values are along the row-axis. Predicted values are along the column-axis. Despite its deeper architecture it makes similar classes of mistakes as our proposed CRNN.

7.4.6. Noise Robustness

Given that we left our raw audio data unchanged we expected a certain degree of noise within the dataset. Therefore we hypothesized that the neural network developed some noise robustness by itself. For instance, the convolution operations of the earlier layers summarize pixel values over an image patch and help with masking noise. To prove our theory we generated two augmented datasets based on the YouTube news dataset. For the first one we mixed the audio signal with randomly generated white noise sound. The resulting audio samples are still easily identifiable for human listeners. The noise has a very strong audible presence, so much so that a human tester will easily be annoyed after a few seconds of listening to it. For the second augmentation we added a more periodic cracking noise emulating analog telephony or a bad voice chat connection. We sampled a selection of fitting sound effects and randomly applied these to the source signal using the PyDub[14] library. The resulting noise is not as noticeable and less intrusive as the white noise, but gives the augmented audio file a subdued vintage quality.

The white noise did deteriorate the language identification performance significantly both for our CRNN proposal and the Inception-v3 CRNN as can be seen in table 11. The noise spectrogram in figure 8 show that the white noise effect has a very strong influence on the resulting image. Most parts of the image end up being covered by the distinct noise texture and only the lower frequency features remain intact. Yet most speech frequencies are still contained in this range. All pauses and fine grained details are also lost to the noise sound. The cracking experiment fared did not incur such a dramatic drop in performance. That might be in part due to the consistent recurring sound of the noise in contrast to the randomly generated white noise sound. Perhaps a second factor was the lower, less intrusive volume used for augmenting this dataset.

The deeper, more complex structure of the Inception-v3 CRNN did suffer a significantly smaller performance deterioration than our proposed CRNN model architecture. The more than five times as many parameters seem to capture the frequency features in a more robust manner. In an attempt to remedy the performance loss for our model we trained and finetuned models containing white noise data. We experimented with a 100% noise dataset and the original news dataset extended with 10% white noise for augmentation purposes. While both approaches did recover some white noise performance they reduced the general language identification and cracking noise statistics as a trade off. Let it be noted that our audio mixing was fully automated and that the resulting samples varied in speech and noise volume. We tried to match volume levels of speech and noise to maintain as much clarity of speech as possible, yet some samples will have an artificial quality to them.

Overall we note that even deep convolutional recurrent networks are still subject to the influence of noise on audio. We learned that our spectrogram preprocessing does not help

[14] `https://github.com/jiaaro/pydub`, accessed 01.03.2017

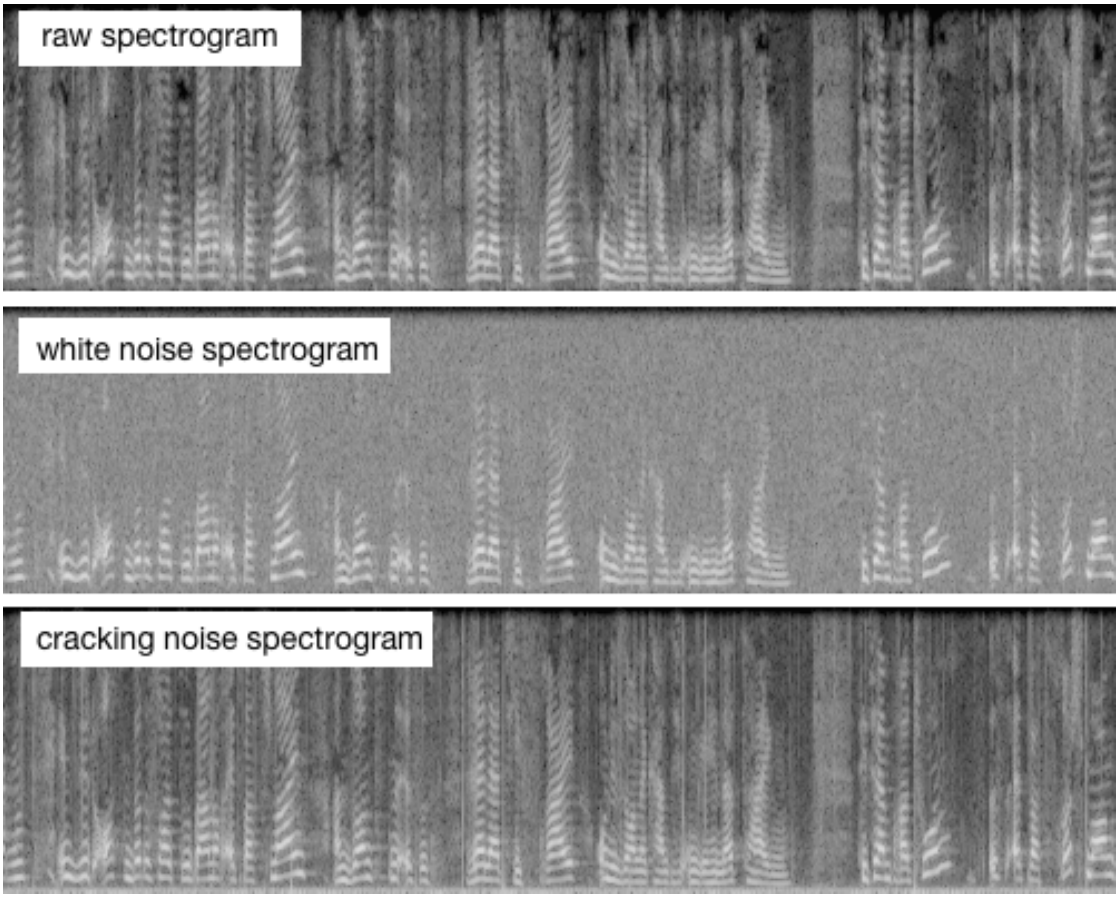

Figure 8: Spectrograms generated from the raw data, augmented with white noise and mixed with a cracking noise emulating analog telephony or a bad voice chat connection. The white noise aggressively subdues most higher frequencies and pauses causing a loss of classification performance. The cracking noise is less intrusive and therefore does not affect accuracy as much.

Dataset	CRNN		Inception-v3 CRNN	
	Accuracy	F1	Accuracy	F1
No Noise	0.91	0.91	0.96	0.96
White Noise	0.63	0.63	0.91	0.91
Cracking Noise	0.82	0.83	0.93	0.93

Table 11: Accuracy and F1 score for our models evaluated on the speech data augmented with two different types of noise. Our proposed model architecture is susceptible to noise and its performance deteriorates. The much deeper architecture of Inception-v3 CRNN is more robust against noise and retains a high performance.

in these situations and that different network architectures play an instrumental role in dealing with these situations.

7.4.7. Background Music Robustness

Many real world audio applications involve some form of music. For example, imagine speaking into your mobile phone from a busy cafe with background music or doing a voice chat session at home while the TV is running. Therefore we evaluated our model to see if language identification was still possible when mixing speech with music. For this experiment we used two different test sets. For the first series we augmented our existing YouTube news dataset with randomly sampled background music similarly to the background noise augmentation. The background audio was obtained from Soundcloud[15] and features royalty free, instrument-only music from various genres, including Pop, Dubstep, Rock and Electro, amongst others. We normalized the volume of these tracks and overlaid them onto our speech data while trying to stay below the speech audio volume. In the best cases the two audio streams blend nicely with the background music producing a soft but noticeable ambient effect, while retaining the clarity of the speech. In some other cases the audio levels and volume are over-accentuate for one or the other. The final result, however, does neither resemble a song nor a piece of music. We changed neither the tempo nor the pitch of our speakers and the rhythm of the vocals does not match the rhythm of the background audio. Therefore we gathered a small second test set of 50 pop and hip hop songs, respectively. Given that our training set does not intentionally include samples with music or songs we expected the performance do be drastically lower then for pure speech samples. Additionally it should be noted that a person's singing voice is different from one's speaking voice.

find good source

Talk with Haojin about omitting the evaluation on songs

[15] `https://soundcloud.com/royalty-free-audio-loops`, accessed 01.03.2017

Dataset	CRNN		Inception-v3 CRNN	
	Accuracy	F1	Accuracy	F1
No Music	0.91	0.91	0.96	0.96
Background Music	0.70	0.70	0.89	0.89
Pop Songs	XX	0.XX	0.26	0.XX
HipHop Songs	XX	0.XX	0.35	0.XX

Table 12: Accuracy and F1 score for our models evaluated on the speech data augmented with background music and songs sampled from four different genres.

7.4.8. Model Extensibility

So far, all our experiments were conducted on datasets consisting of only four languages: English, German, French and Spanish. We extended the existing set with two new languages spoken by millions around the globe: Mandarin Chinese and Russian. The goal of this experiment was to learn whether we could expand our model to other languages. We increased our existing YouTube news dataset with samples taken from Chinese and Russian news channels which now altogether forms the extended YouTube news dataset described in section 5.3. In order to maintain the class distribution for six languages we had to decrease the number of training samples of the existing samples slightly. Table 2 contains the details for the extended YouTube news dataset.

For this experiment we first finetuned our previous best CNN by replacing the final fully connected layers and adjusted the number of output nodes to accommodate for six classes. The resulting model served as the basis for training the CRNN in a similar manner as in earlier experiments. Applied on the test set we measured an accuracy of 0.92 and F1-score of 0.92. Both measurements match our previous evaluation with four languages on the YouTube news dataset as laid out in section 7.4.4 proving that the proposed CRNN architecture can be extended to cover more languages. Figure 9 shows individual performance measures for each language. Mandarin Chinese outperforms all other languages with a top accuracy of 0.96, which could be interpreted as it sounding the most contrasting to western languages and featuring its own unique intonation. We also noted that Russian was most frequently misclassified as Spanish and vice-versa. In contrast to our previous observation German is no longer the worst performing class, but English takes that role now. This is in part due to a significant number of misclassification as Russian samples.

Given that both new languages are rooted within their own respective language families and feature considerable different intonations we were content to find that the features

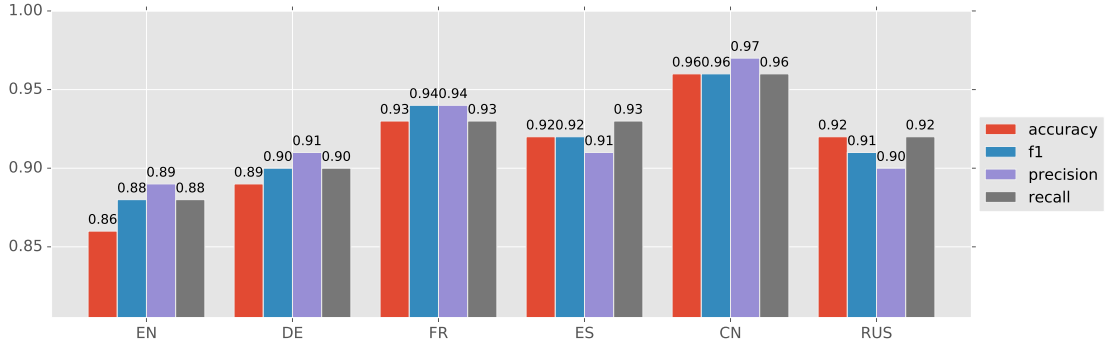

Figure 9: Individual performance measurements for each of our six target languages: English, German, French, Spanish, Mandarin Chinese, and Russian. Chinese exhibits the best misclassification rate while English performs the worst. Overall the model performance is consistent with previous evaluations on four languages as highlighted in section 7.4.4.

learned by our model are indeed universal in nature. We are confident in the believe that the approach to language identification proposed in this thesis can be successfully applied to a wide variety of languages.

7.4.9. Visualizations

All previous sections described and evaluated our model by measuring various performance indicators and applying them to different datasets. In this section we will present plots underlining earlier observations from a visual perspective.

First we visualized the high-dimensional language vector embedding space using the t-distributed stochastic neighbor embedding (t-SNE) algorithm[32]. t-SNE is a nonlinear dimensionality reduction technique employed to map high dimensional data into 2D or 3D space for plotting. We applied this machine learning algorithm to the first 2000 predictions of our second to last fully connected layer right before the classifier and managed to project our 1024 dimensional YouTube news data into a 2D space. Figure 10 shows the resulting plot highlighting a good separation of our four language classes as independent clusters confirming that our network learned effective representations of the audio features. Note that French and Spanish split very nicely while German and English have some overlap. This is in line with our previous observations of classification errors as described in section 5.3.

A primary advantage of deep neural networks is their ability to identify and learn useful features without requiring a data scientist to manually define these. Therefore one does not need any prior knowledge of the domain, a fact that makes these techniques so

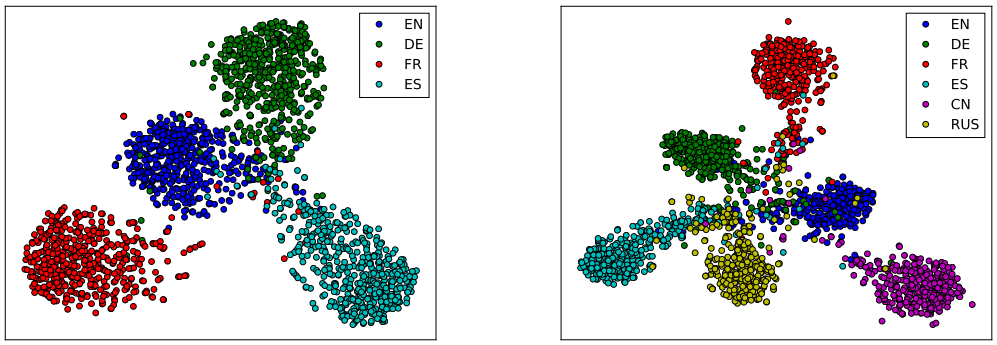

Figure 10: Two-dimensional t-SNE plots of the high dimensional vector representation of our YouTube news samples for a model trained with four and six languages, respectively. All language classes form distinct clusters confirming that the network learned effective representations of the audio features.

powerful and versatile. From an outsider's perspective they can appear as a bit of a black box and it remains unclear which features they ultimately deem relevant. In order to gain a better understanding of our model we visualized its convolutional layers. Figure 11 visualizes nine of the highest activating filters of the final convolutional layer. To obtain these filters we performed back propagation from the output of each filter back to an input image. This yielded the gradients of the output of the filter with respect to the input image pixels. We used that to perform gradient ascent, searching for the image pixels that maximize the output of the filter following the method proposed by Chollet**??**.

Visualizations for the lower-level convolutional layers resulted in images of very simple geometric shapes like lines and circles which matched our expectations. With increasing network depth each convolutional layer's features combined and evolved into more complex shapes resembling our input domain. In figure 11 we can identify the familiar ripple like patterns that form frequency activations over time. This proves that the network learned these structures as we hypothesized earlier. We can also identify that some filters specialize in high frequency whereas others focus on low frequencies. Furthermore, it can be observed that the filters only react to a short and specific span of time within the spectrogram, all of which are less than one second in duration.

7.4.10. Discussion

In this section we introduced and compared different CNN and CRNN architectures for language identification and proved that these models can be used to solve our research

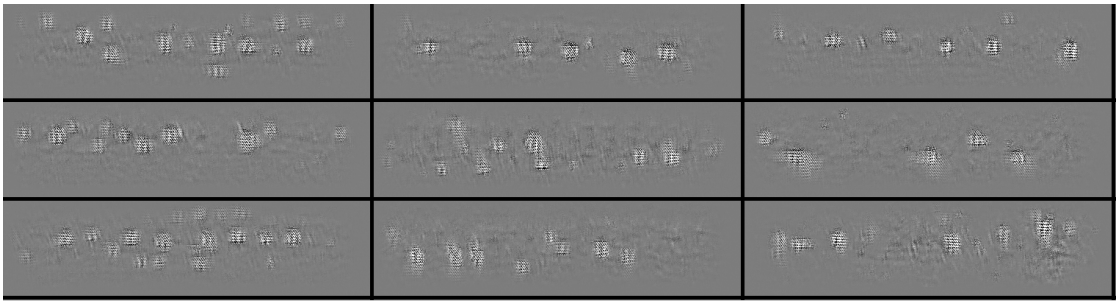

Figure 11: Visualization of nine filters in the final convolutional layer. Note the ripple-like patterns responsible for detecting different frequency spans in the input audio.

task to a satisfying quality. This thesis confirms that audio tasks can be solved within the image domain using spectrogram image representations. As hypothesized we were able to prove that our system learned the audio and frequency structures of these spectrogram images. We showed that these results hold true regardless of the input audio source and across various languages. Further, we demonstrated that the learned intermediate language representations were indeed universal and not language specific and could easily be applied to other languages as well. Our approach of combining convolutional neural networks with bidirectional long short term memory cells showed a consistent improvement over baseline CNNs. In general we were able boost accuracy by at least 1%. Furthermore, we established that an Inception-v3 style CRNN outperformed all other approaches both in terms of accuracy (0.96) as well as with noise robustness.

We presented several augmented datasets to evaluate both noise and background music robustness. We were content to note that the noise resistance of the Inception-v3 CRNN reached acceptable levels without us having to modify or restructure our algorithm. This reaffirms our believe in deep learning techniques as a viable tool for audio tasks.

Our CRNN approach interpreted every vector entry along the x-axis as a separate time step for the recurrent layer. Hence, all results gathered here are representative of only ten seconds of an audio file. We believe that in a production ready system we could increase the prediction performance by doing a majority voting across multiple segments of a longer audio file.

consider do this as part of the demo

Anything else here???

8. Demo Application

- web service - python, react, flask - image - default examples

9. Conclusion and Future Work

9.1. Conclusion

9.2. Future Work

- more languages - different transfer learning: progressive neural networksx - songs - phoneme data - ResNet - LID as distance metric (metric learning) –> enables the use of unknow languages / dialects

Appendices

A. Audio manipulation with SoX

```
scale_factor1=0.94;
scale_factor2=0.05;

sox -m \
    -v $(echo "$(sox input.wav -n stat -v 2>&1) * ${scale_factor1}" |
        bc -l)  input.wav \
    -v ${scale_factor2} <(sox input.wav -p synth whitenoise) \
    -b 16 output.wav
```

Listing 2: Adding white noise to an audio file

B. Background Music Evaluation Sources

English	
HipHop	https://www.youtube.com/watch?v=uelHwf8o7_U&list=RDQMENRLh0JQHf4
Pop	https://www.youtube.com/watch?v=hT_nvWreIhg&list=RDQMxXC2APeBvGc
German	
HipHop	https://www.youtube.com/watch?v=y8pcVDN8vWQ&list=RDQMSi8P9Y1QpqM
Pop	https://www.youtube.com/watch?v=kiMG_JV2gbo&list=RDQM10mSCd8-mJo
French	
HipHop	https://www.youtube.com/watch?v=-n8kGW16RYs&list=RDQMUV6pew7j0sI
Pop	https://www.youtube.com/watch?v=yleB8fUXudw&list=RDQMCawvWO0Heyc
Spanish	
HipHop	https://www.youtube.com/watch?v=CUYrEiymUMY&list=RDQMYSngtE3m2uY
Pop	https://www.youtube.com/watch?v=tIpzfs5tBJU&list=RDQMJhi12p5mKDc

Table 13:

References

[1] Y. K. Muthusamy, E. Barnard, and R. A. Cole, "Reviewing automatic language identification," *IEEE Signal Processing Magazine*, vol. 11, no. 4, pp. 33–41, 1994.

[2] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[3] F. Richardson, D. Reynolds, and N. Dehak, "A unified deep neural network for speaker and language recognition," *arXiv preprint arXiv:1504.00923*, 2015.

[4] F. Richardson, D. Reynolds, and N. Dehak, "Deep neural network approaches to speaker and language recognition," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1671–1675, 2015.

[5] Y. Zhang, E. Chuangsuwanich, and J. R. Glass, "Extracting deep neural network bottleneck features using low-rank matrix factorization.," in *ICASSP*, pp. 185–189, 2014.

[6] O. Plchot, P. Matejka, R. Fér, O. Glembek, O. Novotnỳ, J. Pešán, K. Veselỳ, L. Ondel, M. Karafiát, F. Grézl, *et al.*, "Bat system description for nist lre 2015," in *Proc. Odyssey*, 2016.

[7] A. Sizov, K. A. Lee, and T. Kinnunen, "Discriminating languages in a probabilistic latent subspace,"

[8] R. Zazo, A. Lozano-Diez, and J. Gonzalez-Rodriguez, "Evaluation of an lstm-rnn system in different nist language recognition frameworks," *Odyssey 2016*, pp. 231–236, 2016.

[9] J. Gonzalez-Dominguez, I. Lopez-Moreno, P. J. Moreno, and J. Gonzalez-Rodriguez, "Frame-by-frame language identification in short utterances using deep neural networks," *Neural Networks*, vol. 64, pp. 49–58, 2015.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[11] G. Montavon, "Deep learning for spoken language identification," in *NIPS Workshop on deep learning for speech recognition and related applications*, pp. 1–4, 2009.

[12] H. Kantilaftis, "How to read the news like a professional news anchor." `https://www.nyfa.edu/student-resources/how-to-read-the-news-like-a-professional-news-anchor/`, 2016.

[13] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] E. Jones, T. Oliphant, P. Peterson, *et al.*, "SciPy: Open source scientific tools for Python." `http://www.scipy.org/`, 2001.

[17] R. B. Blackman and J. W. Tukey, "The measurement of power spectra from the point of view of communications engineering - part 1," *Bell Labs Technical Journal*, vol. 37, no. 1, pp. 185–282, 1958.

[18] H. Traunmüller and A. Eriksson, "The frequency range of the voice fundamental in the speech of male and female adults," 1993.

[19] D. Mishkin, N. Sergievskiy, and J. Matas, "Systematic evaluation of cnn advances on the imagenet," *arXiv preprint arXiv:1606.02228*, 2016.

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[21] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[22] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference*, 2014.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[25] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[26] "The 2015 nist language recognition evaluation plan (lre15)." {https://www.nist.gov/sites/default/files/documents/2016/10/06/lre15_evalplan_v23.pdf}„ 2015.

[27] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks.," in *Aistats*, vol. 9, pp. 249–256, 2010.

[29] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[30] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[31] T. Werkmeister and T. Herold, "Practical applications of multimedia retrieval: Language identification in audio files." https://github.com/twerkmeister/iLID/blob/master/Deep%20Audio%20Paper%20Thomas%20Werkmeister%2C%20Tom%20Herold.pdf, 2016.

[32] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.